VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE STŘIHŮ VE VIDEU

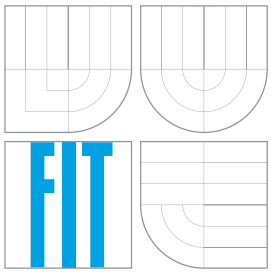
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DUŠAN SUJA

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE STŘIHŮ VE VIDEU

VIDEO SHOT BOUNDARY DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DUŠAN SUJA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL HRADIŠ
SUPERVISOR

BRNO 2010

Abstrakt

Táto práce se zabýva možnostmi detekce střihů ve videu. Z úvodu se věnuje popisu metod pro detekci střihů, jejich výhodám a nevýhodám s ohledem na rychlost a přesnost detekce. V dalších kapitolách je pak popis implementovaných metod pro detekci ostrých střihů pomocí knihovny OpenCV.

Abstract

This thesis is concerned about video cut detections. Introduction explaining methods for video shot cut detecting, their advantages and disadvantages, their speed and accuracy. Next chapters deal with implemented methods for detection of hard cuts using C++ library OpenCV

Klíčová slova

Detekce střihů, video, ostrý střih, OpenCV.

Keywords

Shot cut detection, shot boundary detection, OpenCV.

Citace

Dušan Suja: Detekce střihů ve videu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Detekce střihů ve videu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše

.....
Dušan Suja
18. května 2010

Poděkování

Ďakujem vedúcemu mojej práce Ing. Michalovi Hradišovi za pomoc pri problémoch a za čas, ktorý mi venoval pri riešení tejto práce.

Obsah

1 Úvod **2**

2 Známe metódy detekcie **3**
2.1 Rozdiely intenzity . . . 3
2.2 Porovnanie histogramov . . . 4
2.3 Detekcia hrán . . . 4
2.4 Frekvenčná oblasť . . . 5
2.5 Nastavenie prahu . . . 6

3 Návrh **8**
3.1 Porovnanie po pixeloch . . . 9
3.2 Korelácia . . . 10
3.3 Hrany . . . 11
3.4 Prah . . . 12

4 Implementácia **14**
4.1 OpenCV . . . 14
4.2 Program . . . 15
4.3 Algoritmy . . . 16
4.4 Príprava programu na hromadné testovanie . . . 17

5 Testy **20**
5.1 Popis testovaných dát . . . 20
5.2 Výsledky a porovnanie metód s nastavením pevného prahu . . . 21
5.3 Výsledky a porovnanie metód s adaptívnym nastavením prahu . . . 25

6 Záver **29**

A Obsah CD **32**

B Manuál k programu **33**

C Plagát **35**

.

Kapitola 1

Úvod

V archívoch televíznych štúdií sa od ich vzniku zhromažďuje veľké množstvo multimediálnych dát. Vznikajú rozsiahle videoarchívy. Ako popisuje [15, 4], z masívnym rozšírením videokamier a úložných médií, videoarchívy prestali byť doménou len televíznych štúdií. Veľa nadšencov vlastní doma malý videoarchív svojich výtvorov. Webový videoarchív youtube.com asi prestavovať netreba. Väčšina spoločností, ktoré využívajú videokonferencie sa snáží tieto konferencie zálohovať. Videoarchívy sa začínajú objavovať aj na školách. Záznamy z prednášiek a výukové videá určite pomôžu pri učení. To je len niekoľko príkladov množenia videodát.

Najdôležitejšou požiadavkou na videoarchív je možnosť rýchlo dohľadať potrebné video, prípadne časť videa. Pri hľadaní nás zaujíma obsah videa. Preto je vhodné rozdeliť video na súvislé časti a každej tejto časti priradiť popis. Záznam z koncertu, prednášky, alebo z videokonferencie väčšinou obsahuje jednu ucelenú časť, takže tieto videá sa vyhľadávajú v celosti. Ak máme zostrihané video v televízii alebo v súkromnom videoarchíve, toto video v sebe obsahuje niekoľko častí a pre radenie do archívov je dobré ho rozdeliť na menšie časti.

V zostrihanom videu sa za súvislú časť obsahujúcu ucelenú myšlienku považuje jeden záber. Záber [9] je postupnosť snímok zachytených jednou kamerou v jednom časovom okamihu, bez prerušenia chodu kamery. Jednotlivé zábery sú od seba oddelené strihmi. Strih môže byť ostrý, alebo postupný. Ostrý strih je rozdelenie záberov, ktoré vznikne medzi 2 snímkami. Postupný strih je rozdelenie záberov, ktoré prechádza viacerými snímkami.

Pri vkladaní záznamu do archívu priamo z kamery je rozdelenie na zábery jednoduché. Záznam z kamery obsahuje časové značky, podľa ktorých nieje zložité určiť, kde bolo natáčanie prerušené. Toto prerušenie sa označí ako strih a záznam sa môže ďalej spracovávať. Pri upravení záznamu sa časové značky stratia. Pri archivovaní záznamu tretích strán, prípadne záznamu už spracovávaného, je detekcia strihov komplikovanejšia.

Cieľom tejto práce je porovnať najpoužívanejšie metódy detekcie ostrých strihov a vyhodnotiť výsledky. Na základe výsledkov testovania zostaviť vlastný postup detekcie, prezentovať jeho výsledky a predstaviť jeho výhody a nevýhody.

Práca je rozdelená do niekoľkých kapitol. V úvodnej kapitole (2) popisujem existujúce metódy, ich výhody a nevýhody. V kapitole 3 sa zamýšľam nad výberom vhodných metód, z ktorých budem vychádzať pri návrhu vlastného detektora. Kapitola 4 obsahuje popis implementácie programu a popis použitých knižníc. Testovaniu a vyhodnocovaniu implementovaných metód sa venujem v kapitole 5. Nachádza sa tu aj porovnanie výsledkov testovania.

Kapitola 2

Známe metódy detekcie

Podľa [15] môžeme metódy na detekciu ostrých strihov rozdeliť do dvoch hlavných skupín: metódy priestorových domén a metódy frekvenčných domén. Priestorové domény popisujú porovnávanie na úrovni pixelov, blokov pixelov a snímok. Osobitnú skupinu v priestorových doménach tvorí porovnanie histogramov. Frekvenčné domény popisujú porovnávanie transformačných koeficientov a porovnávanie na úrovni signálov. Transformačné koeficieny sú použíté v [9, 3]. Porovnávanie na úrovni signálov využíva [5], kde sú extrahované príznaky pomocou dekódovania MPEG videa.

Cieľom každého detektoru strihov je identifikovať vlastnosti a charakteristiky snímok pomocou extrahovaných príznakov. Pomocou identifikovaných charakteristík potom rozlíšiť prerušenie postupnosti charakteristík a oddeliť od seba jednotlivé zábery.

Metódy priestorových domén využívajú metrické porovnávanie. To znamená, že na vypočítanie rozdelu medzi snímkami sa využívajú hodnoty intenzity jednotlivých snímok. Najjednoduchšie z týchto metód sú metódy porovnávania 2 nasledujúcich snímok po pixeloch. Medzi ne patria SAD (suma absolútnych rozdielov) a SSD (suma štvorcov rozdielov) ktoré sú bližšie popísané v časti 2.1. Intenzitu snímky je možné vyjadriť aj pomocou histogramu. Porovnaniu histogramov sa venuje sekcia 2.2. Medzi náročnejšie patria metódy založené na detekcii hrán. Popísané su v časti 2.3. Metódy frekvenčných domén sú popísané v časti 2.4.

Výstupom všetkých vyššie spomínaných metód je miera odlišnosti jednej snímky od druhej. Túto mieru je možné vyhodnotiť pomocou nastaveného prahu. Ak je rodiel medzi snímkami väčší ako prahový rozdiel, tak detektor určí, že medzi snímkami je strih. Prah detekcie je možné nastaviť napevno, alebo použiť metódu na jeho automatické nastavenie. Nastavenie prahu je popísané v poslednej časti tejto kapitoly 2.5.

2.1 Rozdiely intenzity

Najjednoduchšou metódou na porovnanie dvoch, po sebe nasledujúcich snímok je suma absolútnych rozdielov intenzity – *SAD* (Sum of Absolute Differences) [13]. Výstupom tejto metódy je

$$SAD = \sum_{i=1}^{n} |x_{1i} - x_{2i}| \tag{2.1}$$

kde x_1 je intenzita pixelu 1. snímky, x_2 je intenzita pixelu 2. snímky a n je počet pixelov v snímke. Táto metóda je jednoduchá a rýchla, preto sa jej modifikácie stali základom pre mnoho porovnávacích metód.

Vyššiu citlivosť na zmenu obrazu má modifikácia zvaná *SSD* (Sum of Square Diferences).

$$SSD = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i}) \cdot (x_{1i} - x_{2i})} \tag{2.2}$$

Nevýhodou týchto metód je vysoká citlivosť na pohyb. Pri pohybe objektov pred kamerou, alebo pri pohybe kamery, sa pixely rovnakej intenzity presúvajú. Porovnanie snímok po jednotlivých pixeloch v tomto prípade vykazuje vysokú mieru odlišnosti. Citlivosť na pohyb sa snaží znížiť metóda *AIM* (Average Intensity Measurement) [15]. Pretože táto modifikácia

$$AIM = |\mu_1 - \mu_2| \tag{2.3}$$

porovnáva priemernú intenzitu pixelov prvej snímky μ_1 s priemernou intenzitou druhej snímky μ_2, pohyb u nej nespôsobuje veľké odlišnosti.

V správach, kde sa často vyskytuje ďalšia obrazovka na pozadí, je časté, že detektor detekuje strih, ktorý vznikne iba v obrazovke na pozadí. To bola príčina vzniku ďalšej modifikácie, ktorá porovnáva intenzitu pixelov po blokoch. Existuje niekoľko variant [1, 15], ale princíp je rovnaký. Snímka sa rozdelí na niekoľko blokov a každý blok sa porovnáva osobitne. Určí sa prah odlišnosti bloku a výstupom je počet blokov, ktoré majú vyššiu mieru odlišnosti ako nastavený prah.

2.2 Porovnanie histogramov

Asi najrýchlejšou metódou porovnania dvoch snímok je *porovnanie histogramov* snímok [7]. Histogram intenzity snímky vyjadruje rozloženie intenzít vrámci snímky. Ak sa objekty v snímke pohybujú, rozloženie intenzít ostáva približne rovnaké. Porovnanie histogramov je preto málo citlivé na pohyb objektov.

Podľa [8] je jednou z možností porovnania výpočet podľa vzorca:

$$HC = \frac{\sum_I (H'_1(I) - H'_2(I))}{\sqrt{\sum_I (H'_1(I)^2) - \sum_I (H'_2(I)^2)}} \tag{2.4}$$

Premenná H_1 označuje histogram vytvorený z prvej porovnávanej snímky. Derivácia $H'_1(I)$ je počítaná podľa vzorca

$$H'_k(I) = \frac{H_k(I) - 1}{N \cdot \sum_J H_k(J)},$$

kde N je počet binov v histograme.

Podobne ako SAD, má aj porovnanie histogramov niekoľko modifikácií, ktoré dokážu v istej miere eliminovať jeho nedostatky.

Nagasaka a Tanaka vo svojej práci [7] popisujú taktiež porovnanie histogramov, ktoré boli vytvorené pre každú farbu zvlášť. Najlepšie výsledky však autori získali pri porovnaní histogramov získaných rozdelením snímok na 16 blokov. Autori práce [1] popisujú ešte niekoľko špecifických modifikácií tejto metódy.

2.3 Detekcia hrán

Účinnejšími, ale náročnejšími metódami porovnávania sú metódy pracujúce s hranami objektov zobrazených na snímke. Základom týchto metód je vyhľadanie všetkých hrán, ktoré sa nachádzajú na snímke.

Ak máme informácie o hranách v 1. snímke, môžeme sa pokúsiť nájsť tie isté hrany aj na nasledujúcej snímke. Podľa toho, koľko hrán z prvej snímky nájdeme v druhej snímke môžeme konštatovať, nakoľko sú snímky podobné/odlišné. V metóde *porovnania hrán* pohyb, ani zmena intenzity nevykazuje vysokú mieru odlišnosti a preto je vhodná na zábery, v ktorých je veľa pohybu. Porovnanie hrán si vybral ako základ svojho detektora Bo Shen [10].

Na základe detekcie hrán a porovnania snímok môžeme určiť vektory pohybov jednotlivých blokov v snímke a kompenzovať pohyb objektov. Vďaka kompenzácii pohybu je možné využiť metódy založené na rodieloch intenzity a potlačiť ich nedostatky spôsobené pohybom objektov alebo kamery.

Ak poznáme, kam sa ktorý blok pohybuje, môžeme pomocou predchádzajúcej snímky a vektorov pohybu zostrojiť očakávanú snímku. Očakávanú snímku porovnáme s aktuálnou snímkou a snímky patriace do jedného záberu budú veľmi podobné, aj keď v nich bol zachytený pohyb. Metóda, ktorá popisuje zostavenie očakávanej snímky sa nazýva *motion prediction*.

2.4 Frekvenčná oblasť

Ak zmeníme pohľad na snímku a predstavíme si ju ako signál, môžeme pomocou korelácie signálu získať ďalšie metódy na detekciu strihov. Korelácia je miera podobnosti dvoch kriviek v závislosti na časovom oneskorení jednej z nich. Bežne sa používa na vyhľadanie krátkeho signálu v dlhom signále. V našom prípade teda môže hľadať časť snímky v celej snímke.

V článku [3] bola použitá metóda *fázovej korelácie PCM*. PCM detekuje pohyb priamo z mapy fázovej korelácie – posun v priestore odráža zmenu fázy. Autori si vo svojej metóde zvolili porovnávanie po blokoch o veľkosti 32 x 32 pixelov. PCM jedného bloku je definované vzorcom

$$p(r_t) = \frac{FT^{-1}\{\widehat{r_t}(\omega)\widehat{r_{t+1}}^{*}(\omega)\}}{\sqrt{\int |\widehat{r_t}(\omega)|^2 d\omega \int |\widehat{r_{t+1}}(\omega)|^2 d\omega}},$$

kde p sú priestorové súradnice vektora a ω sú súradnice frekvenčné. Označenie $\widehat{r_t}(\omega)$ vyjadruje Fourierovú transformáciu bloku r_t, FT^{-1} označuje invertovanú Fourierovú transformáciu a $\{\}^*$ značí komplexné združenie.

Na výpočet korelácie dvoch snímok je možné použíť aj *Pearsonov korelačný koeficient* [13]. Tento korelačný koeficient $\rho_{X,Y}$ medzi dvomi náhodnými hodnotami X a Y s očakávanými hodnotami μ_X a μ_Y a normalizovanými odchýlkami σ_X a σ_Y je definovaný nasledujúcim vzorcom:

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{2.5}$$

E je operátor očakávanej hodnoty. V prípade porovnania 2 snímok môžeme za μ_X a μ_Y dosadiť priemernú hodnotu pixelu v snímke a za σ_X a σ_Y narmalizovanú odchýlku vypočítanu pre všetky pixely snímky.

Bežne komprimované video do MPEG obsahuje P a I snímky.

- I snímka (*intra–frame*) je formátu JPEG a je nezakódovaná. Je to priemerne každá 8. snímka, v závislosti na zmene obsahu nasledujúcich snímok. Ak sa video náhle mení, tak sa pri transformácii videa využíva viac I snímok a video zaberá viac pamäti.

- P snímka (*predicted–frame*) obsahuje len rozdielne hodnoty od I sníky. Takže ak je známa predchádzajúca snímka a rozdiely, je možné zostaviť nasledujúcu snímku.

V [15, 1] autori využili možnosti kompresie videa a do svojich prác vložili aj porovnanie pomocou výpočtu Diskrétnej kosínusovej transformácie obrazu *DCT*. Pre každú snímku (blok snímky) sa získajú koeficienty DCT a porovnajú sa s koeficientami získanými z nasledujúcej snímky. Získanie koeficientov z I snímky a P snímky je rozdielne, preto je k nim potrebné pristupovať odlišne. Pri získavaní koeficientov DCT je problémom to, že ak strih nastane počas P snímok musí sa algoritmus vrátiť na predchádzajúcu I snímku.

Autori práce [15] skombinovali dve metódy porovnávajúce rôzne koeficienty DCT a to tak, že prvá metóda porovnáva len jeden koeficient DCT a druhá – náročnejšia potvrdzuje alebo vyvracia vysledky prvej. Ak prvá metóda detekuje strih a druhá ho nepotvrdí, druhá metóda sa vráti o 30 snímok späť, aby našla I snímku a vďaka nej definitívne potvrdí, alebo vyvráti výsledok prvej metódy.

2.5 Nastavenie prahu

Prvým krokom detektora strihov je zistenie, nakoľko sú dve, po sebe nasledujúce snímky odlišné. V druhom kroku sa pracuje s dátami získanými z prvého kroku a výstupom je konečné rozhodnutie detektora o tom, či sa jedná o strih alebo nie. Postupov na rozhodnutie, či sa jedná alebo nejedná o strih, je niekoľko.

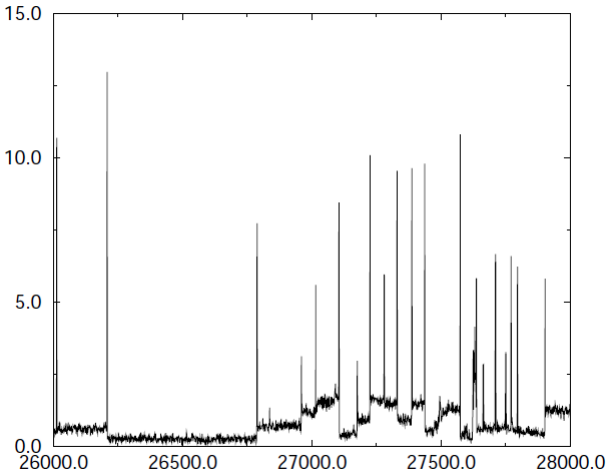
Asi najpoužívaneším postupom je nastavenie pevného prahu. To znamená, že detektoru nastavíme prah citlivosti a on potom vyhodnotí mieru odlišnosti vyššiu ako tento prah ako strih. Nájsť optimálny prah je problém, lebo každé video má inú kvalitu a preto aj iné odlišnosti medzi snímkami. U SAD a SSD je problém aj s rozlíšením videa. Čím má video vyššie rozlíšenie tým je suma rozdielov pixelov vyššia. Toto je možné obmedziť tým, že výsledok SAD/SSD vydelíme počtom pixelov alebo iným zvoleným číslom, ktoré je priamo úmerné počtu pixelov. Asi najpoužívanejším riešením problému s pevným nastavením prahu je možnosť zadať prah citlivosti ako parameter pri spustení programu, prípadne možnosť nastavenia prahu v už spustenom programe.

Pretože na nastavenie prahu je potrebné otestovať, pri akom prahu má detektor najlepšie výsledky a až potom ho nechať detekovať video, začali sa používať iné postupy na nastavenie prahu. V [3] autori použili učiaci sa detektor. Pred detekciou videa detektor spustia v učiacom sa móde a na vstup mu dajú anotované videá. Detektor preskúma videá a podľa anotácii nastaví niekoľko parametrov, ktoré poslúžia na presnejšie detekovanie strihov. Po automatickom nastavení je detektor pripravený detekovať strihy vo videu bez potreby nastavenia prahu pred každou detekciou.

Zábery obvykle vyjadrujú jednu myšlienku. Z toho výplýva, že jeden záber môže mať medzi snímkami minimálne rozdiely, ale ďalší záber môže mať rozdiely medzi snímkami vysoké. Tieto rozdiely je možné vidieť na obrázku 2.1.

Niekedy sa môže dokonca stať, že strih v jednej časti má nižšiu mieru odlišnosti, ako je bežná odlišnosť susedných snímok v inej časti videa. Pritom oproti susedným snímkam ju má niekoľkonásobne vyššiu, takže je možné detekovať, že sa jedná o strih. V [14] autori zvolili adaptívny prah. Adaptívny prah je prah nastavený podľa okolitých snímok. Aby detektor prehlásil skúmaný rozdiel medzi snímkami za strih, musí byť splnených niekoľko podmienok:

- skúmaný rozdiel je maximálny z okolitých m testovaných rozdielov



Obrázek 2.1: Miera odlišnosti medzi dvomi po sebe nasledujúcimi snímkami v závislosti na pozícii vo videu (*obrázok prevzatý z [14]*)

- skúmaný rozdiel je n krát vyšší ako maximum okolitých testovaných rozdelov.

Autori v práci [14] testovali tri modely nastavenia prahu, ktoré vychádzajú z rovnakého základu. Najprv je potrebné si určiť počet snímok po strane, voči ktorým chceme nastaviť prah. Ak počet snímok po strane bude m, tak miera odlišnosti v strede skúmaných snímok musí byť maximum sledovaných snímok v rozsahu 2m-1. Skúmaná miera odlišnosti musí byť väčšia ako

$$max(\mu_{left} + T_d\sqrt{\sigma_{left}}, \mu_{right} + T_d\sqrt{\sigma_{right}}) \tag{2.6}$$

, kde za T_d dosadili autori hodnotu 5. Podľa autorov má zvolená metóda lepšie vysledky ako vyššie spomínaná všeobecná metóda, lebo počíta so strednou hodnotou a odchýlkou. Vďaka tomu dokáže dobre detekovať strih aj tam, kde je prechod medzi záberom s malými odlišnosťami a záberom s veľkými odlišnosťami.

Z podobného základu vychádzali aj autori [12], ale detekcia strihu/nestrihu bola vložená priamo do prvého kroku detektora, v ktorom detektor zisťuje mieru odlišnosti. V práci je implementovaný pohľad na 10 snímok naraz a počítanie miery odlišnosti je realizované v závislosti na ostatných snímkach.

Kapitola 3

Návrh

Metód na detetekciu strihov vo videu je mnoho a vďaka tomu, že väčšinu z nich už niekto otestoval a zverejnil výsledky testov, sa výber vhodnej metódy zjednodušil. Dôležité je zvoliť si tú správnu, prípane si ju správne upraviť. Každá metóda ma svoje výhody a nevýhody.

Vďaka tomu, že niektoré nevýhody metód je možné eliminovať spojením s inou metódou, výber už nespočíva len v nájdení tej správnej metódy, ale hlavne v ich správnom skombinovaní. Z dostupných materiálov je možné vybrať metódu podľa popísaných vlastností a detektor zostaviť tak, že bude mať slušné vysledky v širokej škále videí. Zrejme však nieje reálne vziať do úvahy všetky možné problémy detekovania strihov a tieto dokonale odstrániť v jedinom detektore bez nastavenie dodatočných parametrov. Preto je možné zostaviť detektor, ktorému nastavíme o aký druh videa sa jedná a vďaka tomu si môže z implementovaných metód zvoliť najvhodnejšie metódy pre daný druh videa.

Pri výbere vhodných metód pre môj detektor som vychádzal z poznatkov získaných z priloženej literatúry. Vďaka jednoduchosti základných metód porovnávajúcich rozdiely intenzít pixelov som ako prvé implementoval tieto metódy. Bližši popis výberu metód z tejto kategórie je popísaný v sekcii 3.1.

Histogramy sú v literatúre popisované ako veľmi rýchle algoritmy. Ich ryclosť a dobré vlastnosti pri pohyblivom videu sú uvádzané ako veľká výhoda oproti ostatným algoritmom. Z týchto dôvodov som sa rozhodol do svojej práce zahrnúť aj algoritmy pracujúce s histogramami. Zvolené histogramy, spôsob ich vloženia do metódy porovnania histogramov a dôvody výberu sú taktiež popísane v časti 3.1.

Do svojho detektoru som sa rozhodol pridať aj metódu využívajúcu koreláciu pixelov. Bližší popis metódy je v sekcii 3.2.

Myslím, že pre túto prácu nieje nevyhnutné zasahovanie do kompresie videa, preto som v detektore vynechal frekvenčné metódy.

Aby implementácie detektora nebola príliš jednoduchá na úroveň bakalárskej práce, rozhodol som sa implementovať aj metódy, používajúce detekciu hrán. Detekcia hrán je náročná, ale určite bude mať dobré výsledky. Aby bola detekcia hrán dobre využitá, spojil som ju s niekoľkými základnými porovnávacími metódami. Popis detekcie a následného spojenia je podrobnejšie rozpísaný v časti 3.3.

Prah nastavený napevno sa vyskytuje vo väčšine prác, popisujúcich detektory, preto som sa rozhodol pre implementáciu pevného prahu. Automaticky nastevený prah citlivosti má určite svoje výhody, preto som sa rozhodol do svojho detektora zahrnúť aj túto možnosť. Detajly nastavenia prahu popisujem v sekcii 3.4.

3.1 Porovnanie po pixeloch

Ako prvú a základnú metódu svojho detektora som zvolil metódu SAD. Výstupy tejto metódy poslúžia ako základný prvok, voči ktorému budem porovnávať ostatné metódy.

Aby som zaručil podobné vústupy u videí z rôznym rozlíšením, upravil som pôvodný vzorec 3.1 nasledovne:

$$SAD = \frac{5}{n}\sum_{i=1}^{n}[|x_{1ir} - x_{2ir}| + |x_{1ig} - x_{2ig}| + |x_{1ib} - x_{2ib}|] \tag{3.1}$$

Rozdiely intenzít som spočítal pre každú farbu osobitne. x_{1ir} vo vzorci vyjadruje intenzitu červeného pixelu s indexom i z prvej porovnávanej snímky. Indexy r, g, b pri hodnotách x_1 a x_2 označujú farby porovnávaných pixelov.

Aby výstup nebol závislý na rozlíšení videa, tak som sa rozhodol vydeliť sumu rozdielov počtom pixelov n. Pre zvýšenie presnosti výstupu, ktorý je celočíselný, som sa rozhodol nedeliť sumu celým počtom pixelov, ale len jeho $\frac{1}{5}$, a tým 5 – násobne zvýšiť presnosť.

Metóda SSD má vyššiu citlivosť, preto som sa rozhodol implementovať aj túto metódu. Výpočet tejto metódy som si upravil podobne ako u metódy SAD. Aby som zachoval správny pomer k počtu pixelov, zvolil som si v tomto prípade v menovateli odmocninu z počtu pixelov namiesto vydelenia počtu pixelov konštantou 5. Odmocninu som si do menovateľa zvolil preto, lebo sa vyskutuje aj v čitateli. Podobne ako v predchádzajúcej metóde som aj tu zvolil súčet počítaný osobitne pre všetky farby. Upravený vzorec 2.2 vyrezá takto:

$$SSD = \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^{n}[(x_{1ir} - x_{2ir})^2 + (x_{1ig} - x_{2ig})^2 + (x_{1ib} - x_{2ib})^2]} \tag{3.2}$$

Ako ďalšiu metódu som si zvolil AIM. Táto metóda čiastočne odstraňuje nevýhodu SAD a SSD, a to veľkú citlivosť na pohyb. Keďže AIM počíta priemernú intenzitu pixelu v jednej snímke a priemernú intenzitu pixelu v druhej snímke, neje potrebné vzorec upravovať kvôli rozdielnemu počtu pixelov vo videách. Jedinú úpravu ktorú som vo vzorci 2.3 spravil bola úprava na počítanie priemernej intenzity pre každú farbu. Po upravení pre farby vznikne vzorec:

$$AIM = \sum_{c=1}^{3}|\mu_{1c} - \mu_{2c}| = \sum_{c=1}^{3}\left|\frac{\sum_{i=1}^{n} x_{1ic}}{n} - \frac{\sum_{i=1}^{n} x_{2ic}}{n}\right| \tag{3.3}$$

kde c je index farby: červená=1, zelená=2 a modrá=3. Hodnota prvku x_{1ic} vyjadruje teda intezitu farby c pixelu x, ktorý sa nachádza pod inexom i na prvej porovnávanej snímke.

Pre rýchlosť a dobré vlastnosti histogramov som sa rozhodol histogramy zahrnúť do svojej práce. Histogram obsahuje informácie o tom, koľko pixelov je v daných rozsahoch intenzít. Takže ak sa vo videu objekt pohybuje, mení sa síce poloha pixelov, ktoré ho znázorňujú, nemení sa však ich rozsah. Na základnú metódu porovnania 2 histogramov existuje niekoľko modifikácií.

Pri výbere správnej modifikácie pre môj detektor som zvažoval histogram zo šedotónového obrázku, histogramy pre všetky farby a histogramy z blokov obrázku. Porovnanie histogramov extrahovaných z obrázku v stupňoch šedi je úplne najzákladnejšia metóda, a preto si myslím, že bude mať aj najslabšie výsledky. Podľa môjho názoru, ak rozdelím snímku na malé bloky a z každého tohto bloku budem počítať histogram, metóda stratí svoju výhodu malej citlivosti na pohyb. Ak vytvorím histogram z každej farby osobitne a

výsledky sčítam, strojnásobím citlivosť detekcie oproti porovnaniu histogramov extrahovaných zo snímok skonvertovaných do stupňov šedi.

Z vyššie uvedených dôvodov som si zvolil modifikáciu, pri ktorej extrahujem histogramy jednotlivých farieb a vypočítam rozdiel histogramov pre každú farbu osobitne. Výstupom metódy bude teda súčet rozdielov každej farby. Pre výpočet s viacerými farbami som upravil vzorec 2.4 nasledovne:

$$HC = \sum_{c=1}^{3} \left(1 - \frac{\sum_I (H'_{1c}(I) - H'_{2c}(I))}{\sqrt{\sum_I (H'_{1c}(I)^2) - \sum_I (H'_{2c}(I)^2)}} \right) \cdot 1000 \tag{3.4}$$

Vzorec je upravený tak, aby na výstupe bola namiesto miery podobnosti, miera odlišnosti. Premenná H_{1c} označuje histogram vytvorený z prvej porovnávanej snímky a z farby c. Výsledkom korelácie 2 histogramov je číslo v intervale $\langle -1, 1 \rangle$. Ak sú histogramy rovnaké, tak výsledkom bude 1, ak sú úplne odlišné, tak bude výsledkom -1. Aby mi metóda vracala hodnotu, vyjadrujúcu mieru odlišnosti, tak koreláciu odčítam od čísla 1. Pretože výstupom metódy má byť celé číslo, výstupom bude 1000 násobok tohto odčítania. Vďaka tomu dostanem na výstupe pri porovnaní rovnakých histogramov 0. Pri porovnaní úplne odlišných histogramov všetkých farieb bude celková výstupná hodnota 6000.

Aby som zúžitkoval svoje poznatky, pridal som do detektora aj vlastnú metódu založenú na porovnaní pixelov. Rozdelil som snímku na bloky s veľkosťou 32×32 pixelov. Výsledkom metódy BX je suma odlišností jednotlivých blokov.

$$BX = \sum_{i=1}^{m} BLOC_i \tag{3.5}$$

Premenná m označuje počet blokov v snímke. Aby som čo najviac zvýšil citlivosť na zmenu obrazu, rozhodol som sa do celkovej sumy pričítať sumu bloku len za určitých podmienok. Mieru odlišnosti bloku preto pričítavam k celkovej miere len v prípade, že suma rozdielov intenzít pixelov v bloku je vyššia, ako počet pixelov v celej snímke. Takže hodnotu $BLOC$ je možné dostať po dosadení odlišnosti do

$$BLOC = \begin{cases} 0, & d(X) < n \\ \frac{d(X)}{n}, & d(X) > n \end{cases}$$

kde n je počet pixelov v celej snímke. Samotnú mieru odlišnosti $d(X)$ počítam porovnaním každého pixelu bloku s pixelom totožného bloku druhej snímky. Aby som znížil citlivosť na pohyb, porovnávam aj 8 – okolie pixelu. Vytvorený vzorec preto môže vyzerať podobne:

$$d(X) = \left| \sum_{x=0}^{31} \sum_{y=0}^{31} \left(\sum_{i=-1}^{1} \sum_{j=-1}^{1} \left(X_{1(x,y)} - X_{2(x+i,y+j)} \right) + \left(X_{1(x,y)} - X_{2(x,y)} \right) \right) \right|$$

Súradnice pixelu v bloku sú vyjadrené premennými x a y. Kedže blok má veľkosť 32×32 pixelov, maximálna hodnota súradnice x alebo y je 31. Aby mal rozdiel priamych pixelov pri porovnávaní vyššiu mieru ako rozdiel okolitých pixelov, k sume pričítavam tento rozdiel ešte raz.

3.2 Korelácia

Z metód založených na frekvenčných doménach som si pre jednoduchosť vybral metódu krížovej korelácie, s použitím Pearsonovho korelačného koeficientu. Pre lepšiu citlivosť som

si aj túto metódu upravil tak, aby počítala koreláciu pre každú farbu osobitne.

Najprv si pre každú farbu každej snímky vypočítam priemernú hodnotu a normalizovanú odchýlku. Potom vypočítam pomocou premernej hodnoty očakávanú hodnotu a túto vydelím súčinom odchýliek.

K vzorcu na výpočet korelácie 2.5 doplním rozdelenie pre farby. Koraláciou dvoch signálov dostanem na výstupe 1 ak sú signály zhodné a -1 ak sú signály úplne rozdielne. Aby som túto funkčnosť mohol vložiť do môjho detektora, potreboval som upraviť mieru podobnosti na mieru odlišnosti podobne ako pri korelácii histogramov. Preto som pridal do vzorca odčítanie korelácie od 1. Na koniec ešte celkový výsledok vynásobím 1000, aby som zaručil presnosť pri prevedení desatinného čísla na celé. Po upravení vzorca 2.5 som dostal vzorec

$$CrosCorrel = \sum_{c=0}^{3} \left(1 - \frac{E[(x_{1ic} - \mu_{1c})(x_{2ic} - \mu_{2c})]}{\sigma_{1c}\sigma_{2c}}\right) \cdot 1000 \tag{3.6}$$

kde c vyjadruje farbu, μ_i značí priemernú hodnotu snímky i a σ_i označuje normalizovanú odchýlku vypočítanú zo snímky i. Očakávanú hodnotu E dostaneme podľa vzorca

$$E[(x_{1ic} - \mu_{1c})(x_{2ic} - \mu_{2c})] = \frac{1}{n}\sum_{i=1}^{n}(x_{1ic} - \mu_{1c})(x_{2ic} - \mu_{2c}),$$

v ktorom n označuje počet pixelov v snímke.

3.3 Hrany

Do môjho detektoru som sa rozhodol implementovať aj metódu využívajúcu hrany objektov v snímkach. V oboch snímkach nájdem hrany a snažím sa určiť, kam sa ktorý objekt posunie. Keď viem, kam sa objekt posunie, tak môžem vytvoriť predpokladanú snímku a nasledujúcu snímku porovnať s touto predpokladanou. Pri tomto porovnávaní však nieje potrebné zostavovať predpokladanú snímku, stačí len nasledovnú snímku porovnať s predchádzajúcou a do porovnania zahrnúť kompenzáciu pohybu.

Výstupom tejto metódy bude informácia o tom, koľko sa ktorý pixel z prvej snímky presunul do druhej snímky : Motion Estimation(ME). Takže každému pixelu z prvej snímky pribudne vektor pohybu do druhej snímky. Porovnanie vektorov samotných nemá veľký zmysel, preto sa metódy detekujúce hrany spájajú s metódami metrickými. O porovnávanie s predpokladaným obrázkom som preto rozšíril niektoré vyššie spomenuté metódy.

Asi najvhodnejšie na rozšírenie o pohyb objektov sú metódy SAD a SSD. Zahrnutím kompenzácie pohybu do metódy SAD vznikol upravený vzorec na:

$$SADME = \frac{5}{n}\sum_{i=1}^{n}[|x_{1ir} - x_{2(i+v)r}| + |x_{1ig} - x_{2(i+v)g}| + |x_{1ib} - x_{2(i+v)b}|] \tag{3.7}$$

kde k pôvodnému vzorcu pribudla premenná v ktorá vujadruje vektor pohybu daného pixelu. Podobne bolo treba upraviť aj vzorec pre metódu SSD:

$$SSDME = \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^{n}[(x_{1ir} - x_{2(i+v)r})^2 + (x_{1ig} - x_{2(i+v)g})^2 + (x_{1ib} - x_{2(i+v)b})^2]} \tag{3.8}$$

Keďže metóda AIM neporovnáva priamo pixel s pixelom a pracuje so snímkou ako celkom, nemá zmysel ju rozširovať o odhad pohybu. Pri počítaní histogramov so snímok

tiež pracujem so snímkou ako celkom. To že sa nejaký objekt pohne alebo nepohne, nieje pri histograme dôležité. Preto do metódy porovnania histogramov tiež nemá zmysel vkladať možnosť porovnania snímok upravených podľa odhadu pohybov.

Moja vlastná metóda, ktorá počíta s blokmi pixelov má zmiernenú citlivosť na pohyb vďaka porovnaniu okolia pixelu. Som však presvedčený, že keď do porovnania pridám kompenzáciu pohybu, výsledky tejto metódy sa zlepšia. Po vložní vektorov pohybu môžeme základný vzorec tejto metódy zapísať nasledovne:

$$d_{ME}(x) = \left| \sum_{i=0}^{B} \sum_{j=-5}^{5} \left(I_{2(i+v)} - I_{1(i+j)} \right) \right| \tag{3.9}$$

Ostatné časti pôvodného vzorca ostanú nezmenené.

Korelácia vyjadruje podobnosť jedného signálu k druhému. Ak pohyb objektov vykompenzujem, myslím, že podobnosť bude vyššia a vďaka tomu aj korelácia bude vykazovať lepšie výsledky. Vďaka nepresnosti kompenzácie pohybu však môže dôjsť k väčším odlišnostiam ako v prípade, nepoužitia kompenzácie. Rozhodol som sa však kompenzáciu pohybu implementovať aj do metódy krížovej korelácie. Po pridaní kompenzácie pohybu bude vzorec 3.6 upravený na

$$CrosCorrelME = \sum_{c=0}^{3} \left| \frac{E[(x_{1ic} - \mu_{1c})(x_{2(i+v)c} - \mu_{2c})]}{\sigma_{1c}\sigma_{2c}} - 1 \right| \cdot 1000 \tag{3.10}$$

Výpočet priemernej hodnoty a odchýlky sa nezmení. Zmení sa jedine výpočet očakávanej hodnoty, kde sa pixel z prvej porovnávanej snímky s indexom i porovná s pixelom druhej snímky, ktorý je na indexe vypočítanom pomocou vektorov.

3.4 Prah

Detektor by mal obsahovať možnosť nastavenia prahu. Preto som sa rozhodol za výpočet rozdielu vložiť ďalší prvok, ktorý zaistí porovnanie rozdielu so zadaným prahom a vyhodnotí rozdiel medzi snímkami za strih, alebo nestrih. Do tohto prvku som implementoval metódu na porovnanie rozdielu s pevne nastaveným prahom a metódu na porovnanie rozdielu s dynamicky nastaveným prahom.

Porovnanie s pevným prahom je veľmi jednoduché. Túto funkčnosť som zahrnul priamo do procesu porovnávania snímok. To znamená, že detektor vyhodnotí rozdiel medzi snímkami a hneď tetno rozdiel porovná so zadaným prahom. Detektor s pevným prahom nepotrebuje skákať medzi snímkami, a preto mu postačuje jeden prechod celým videom.

Porovnanie s dynamicky nastaveným prahom je už komplikovanejšie. Najprv je potrebné vyhodnotiť rozdiely snímok cez celé video a potom je možné dynamicky nastavovať prah. V prvom kroku teda použijem jednu z vyššie spomenutých metód. Prejdem celé video a pre každý prechod dvoch snímok vyhodnotím mieru odlišnosti. V druhom kroku si vezmem miery odlišností okolitých snímok a vyhodnotím, nakoľko práve sledovaná odlišnosť zapadá do postupnosti okolitých odlišností. To znamená, že nastavím prah podla okolitých snímok a porovnám sledovanú odlišnosť s týmto prahom. Pre rozhodnutie, či sa jedná, alebo nejedná o strih potom postačuje výsledky vypočítať podľa vzorca

$$CUT = \begin{cases} 0, & d(x) \leq max(\mu_{left} + T_d\sqrt{\sigma_{left}}, \mu_{right} + T_d\sqrt{\sigma_{right}}) \\ 1, & d(x) > max(\mu_{left} + T_d\sqrt{\sigma_{left}}, \mu_{right} + T_d\sqrt{\sigma_{right}}) \end{cases} \tag{3.11}$$

Kde $d(x)$ predstavuje práve sledovanú mieru odlišnosti vypočítanu v prvom kroku detektora. Hodnoty μ a σ je počítaná z piatich prechodov pred aktuálnym prechodom a piatich prechodov za aktuálnym. Premennú T_d je možné nastaviť ako parameter programu, aby som mohol otestovať, kedy dosahuje najlepšie výsledky. Pred aplikovaním tohoto vzorca najprv zistím, či práve sledovaný rozdiel je najvyšší spomedzi okolitých rozdielov zahrnutých do vzorca. Ak nieje najvyšší, tak jednoducho prejdem ďalej bez nutnosti dosadzovania do vzorca a rozdiel vyhodnotím ako nestrih.

Kapitola 4

Implementácia

Aby som vyhodnotil a porovnal vlastnosti jednotlivých metód, vytvoril som vlastný detektor strihov. Tento dektektor som implementoval v programovacom jazyku C++. Pre prácu s videom som využil open–source knižnicu OpenCV [8], ktorá je napísaná v jazyku C/C++. Do detektoru som zahrnul všetky postupy popísané v predchádzajúcej kapitole. Detektor som vyvýjal pod operačným systémom Linux. Zdrojový kód je prenositeľný, takže detektor je možné spustiť aj pod operačným systémom Windows.

Na začiatku tejto kapitoly som popísal časti knižnice OpenCV, ktoré som použil v mojej práci. Ďalej sú bližšie popísané jednotlivé implenetované časti detektora. Následne je podrobne popísaná implementácia konkrétnych algoritmov. V poslednej časti tejto kapitoly je popísaná úprava programu na hromadné testovanie algoritmov.

4.1 OpenCV

Knižnica OpenCV (Open Source Computer Vision) [8, 2] ponúka mnoho funkcií a dátových typov na prácu s obrázkami a videom. Je zameraná najmä na počítačové videnie a spracovanie obrazu v reálnom čase. Túto knižnicu je možné využiť v programoch napísaných v jazyku C a C++, prípadne s pridaným rozhraním v jazyku Python a Octave. OpenCV je k dispozícii pre operačné systémy Windows, Linux a Mac OS.

Použitie tejto knižnice značne zjednodušuje prácu s videom. Programátora nemusí zaujímať spracovanie obrazu na najnižšej úrovni a môže sa plne venovať implementácii navrhnutých algoritmov.

Detektor pracuje s videom, takže aby mohlo byť video načítané, musia byť v systéme nainštalované správne kodeky. Knižnica OpenCV poskytuje na načítanie videa funkciu `cvCreateFileCapture()`. Táto funkcia načíta zadané video a vráti ukazovateľ na toto video vo formáte `CvCapture *`. Z tohto dátového typu je možné získať niekoľko informácii o videu pomocou metódy `cvGetCaptureProperty()`. Najdôležitejšie sú asi výška a šírka videa, informácia o počte snímok za sekundu a celkový počet snímok.

Na porovnanie potrebuje detektor pristupovať k jednotlivým snímkam ako k obrázkom. K tomu slúži funkcia `cvQueryFrame()`, ktorá vráti ukazovateľ na obrázok nasledujúcej snímky, vo formáte `IplImage *`. Objekt `IplImage` umožnuje pristupovať k jednotlivým farbám pixelov ako k trojrozmernej matici s rozmermi $x \times y \times k$, kde x a y sú súradnice pixelu v snímke a k je index farby.

Na zobrazenie obrázku slúži metóda `cvShowImage()`, ktorá obrázok zobrazí v otvorenom zobrazovacom okne. Toto okno na zobrazenie obrázku je možné otvoriť pomocou funkcie

`cvNamedWindow()`, ktorej sa ako parametre predajú názov a rozmery okna.

OpenCV obsahuje mnoho užitočných metód a dátových typov, ktoré sú využiteľné v špeciálnych prípadoch. Okrem vyššie spomínaných som špeciálne metódy a dátové typy knižnice OpenCV využil v metóde na porovnanie histogramov a pri počítaní kompenzácie pohybu. Tieto sú bližšie popísané v príslušných sekciách.

4.2 Program

Program je možné spustiť s niekoľkými parametrami, bližšie popísanými v dodatku B. Základnými parametrami programu sú názov vstupného videa a funkcia, ktorá sa má spustiť. Ostatné parametry pribudli pre potrebu testovania programu.

Vstupným bodom programu je súbor `Main.cpp`. V ňom sa nachádza kontrola parametrov a následný výpis chybových hlášiek. Je tu tiež rozpísaná nápoveda k parametrom programu.

Základom celého detektora je trieda `Videocapture`. V tejto triede sa nachádza celá logika detektora. Trieda sa skladá zo štyroch hlavných častí.

Prvá časť obsahuje kontroly na vstupné súbory, ktoré sa volajú z hlavného bodu programu pri kontrole parametrov. V tejto časti sa inicializujú všetky potrebné dátové typy a alokuje sa potrebné miesto v pamäti.

Po skončení prvej časti sa chod programu dostane k vlastnému vyhodnocovaniu prechodov snímok vo videu. Podľa zadaných parametrov sa určí, ktoré metódy sa majú použiť. Toto je časovo najnáročnejšia časť programu.

Keď detektor dokončí spracovávanie videa, dostane sa do tretej časti tejto triedy. V tejto časti je implementovaná logika, ktorá podľa rozdielov zistených v druhej časti rozlišuje, či sa jedná o strih, alebo nie. V prípade použitia detekcie s pevným prahom je táto funkčnosť vykonávaná už počas počítania rozdielov medzi snímkami.

Po dokončení detekcie sa podľa zvolených parametrov ďalšia činnosť predá buď do prehrávača videa, alebo do triedy `Sim`. V prípade voľby prehratia videa sa program presunie do štvrtej časti triedy. Znovu načíta video a postupne sa začnú v otvorenom okne zobrazovať dvojice snímok. Ak dôjde k zobrazeniu snímok, medzi ktorými je strih, prehrávanie videa sa zastaví. Tým je možné vizuálne overiť správnosť detekovaných strihov, prípadne zistiť, kde detektor zlyhal.

Pri ukončení práce triedy sa spustí čistenie programu. Všetky inicializované premenné sa odstránia z pamäti a alokované miesto sa uvoľní.

Ďalšou časťou programu je trieda `Sim`, ktorá sa stará o priebeh testovania detektora. Táto trieda je bližšie popísaná v sekcii 4.4.

Aby som zorganizoval všetky súbory potrebné pre chod programu, vytvoril som pre program adresárovú štruktúru. Hlavná zložka, v ktorej je program obsahuje šesť podzložiek. Zdrojové súbory programu sa nachádzajú v zložke src. Súbory potrebné pre vytvorenie dokumentácie k bakalárskej práci sa nachádzajú v zložke doc. Testovacie videá, je potrebné umiestniť v zložke video a anotácie k týmto vydeám musia byť v zložke anotation. Do zložky myout sa ukladajú dočasné súbory, počas behu detektora. Všetky vypočítané štatistické hodnoty sa ukladajú to príslušných súborov v zložke plot.

4.3 Algoritmy

Prvým a základným algoritmom, ktorý som implementoval bol algoritmus metódy SAD. Pri implementácii som vychádzal zo vzorca 3.1 s tým, že v cykle prechádzam všetky pixely a k sume pričítam rozdiel intenzít jednotlivých farieb týchto pixelov. Pre každú farbu som si vytvoril osobitnú sumu. Po skončení cyklu sčítam absolútne hodnoty týchto súm a vydelím počtom pixelov. Hodnoty na výstupe metódy sa pohybujú v rozmedzí 0 – 2000.

Pre porovnanie som ako druhú implementoval metódu SSD. Postupoval som podobne ako pri implementácii SAD, ale vychádzal som zo vzorca 3.2. Vďaka odmocnine vo vzorci má metóda výstup v menšom rozmedzí ako metóda SAD. Na výstupe je možné dostať hodnoty od 0 do 300.

Nasledujúcim algoritmom, ktorý som vložil do svojho detektora bol algoritmus metódy AIM. Pri implementácii som vychádzal zo vzorca 3.3 s tým, že podobne ako pri SAD, v cykle prechádzam všetky pixely a z každého pixelu osobitne zisťujem intenzitu jednotlivých farieb. Pre každú farbu a každú snímku je vytvorená osobitná suma a v metóde sa ku každej sume pričítava hodnota intenzity odpovedajúcej farby každého pixelu. Na konci metódy je každá s týchto súm vydelená počtom pixelov z čoho vznikne priemerná hodnota každej farby oboch snímok. To znamená, že ak má video 3 farby, tak vznikne 6 súm. Priemerné hodnoty odpovedajúcich farieb prvej a druhej snímky sú od seba odčítané. Absolútne hodnoty týchto rozdielov su sčítane. Metóda vracia celočíselné hodnoty v rozmedzí 0 – 400.

Ako ďalšiu v poradí som implementoval vlastnú metódu, ktorú som nazval BX. Pri implementácii som vychádzal zo spomínanej rovnice 3.5. Pri porovnávani pixelov v bloku (3. časť vyššie spomínanej rovnice) som postupoval nasledovne:

- do sumy bloku som 2 x pričítal výsledok odčítania intenzity pixelu zo snímky 1 od intenzity pixelu zo snímky 2: $(2 \cdot (I_{2i} - I_{1i}))$
- ďalej som do sumy pričítal rozdiely intenzít pixelov v okolí porovnávaneho pixelu na snímke 2 od intenzity porovnávaneho pixelu na snímke 1
- podobne som postupoval pre všetky pixely v bloku.

Aby som zachoval vyššiu citlivosť, tento postup som realizoval pre každú farbu osobitne. Výstup metóty je v rozmedzí 0 – 2000

Poslednou zo základných implementovaných metód a zároveň najrýchlejšou bola metóda porvnania histogramov. Pri implementácii HC som okrem metódy `cvCreateImage` na vytvorenie obrázku využíl aj metódu `cvCreateHist` ktorá vytvorí histogram. Na výpočet histogramu som použil ďalšiu metódu z knižnice OpenCV a to metódu `cvCalcHist`. Pretože táto metóda pracuje priamo z obrázkom, rozhodol som sa neukladať každú snímku do pamäti, ale do pamati uložiť priamo vypočítaný histogram. Preto sa metóda pri prvom volaní inicializuje, kedy vytvorí histogram z aktuálne načítanej snímky, ale neoprovnáva. Metóda začne histogramy porovnávať až pri ďalšom volaní, kedy je už v pamäti uložený predchádzajúci histogram. Základ vzorca 3.4, ktorý som spomínal vyššie, je priamo implementovaný vo funkcii `cvCalcHist`. Takže som len k výsledku pridal prevod z podobnosti na odlišnosť, ktorý som bližšie popísal v sekcii 3.1. Výstupom tejto metódy je 0 pri porovnaní rovnakých snímok a 6000 pri porovnaní maximálne odlišných snímok.

Po naprogramovaní týchto metód som pridal možnosť automatického nastavenia prahu. Ak bol zadaný parameter pre výpočet s pevným prahom, rozhodovanie detektora o strihu je vložené do hlavného cyklu spracujúceho celé video. Ak bol zvolený výpočet strihov s automaticky nastaveným prahom, rozhodovanie o strihu sa presúva až na koniec programu. Do

detektora som implementoval funkčnosť podľa vzorca 3.11. Na výpočty s rozdielmi snímok pred aktuálnym a rozdielmi snímok za aktuálnym rozdielom som využil dva akumulátory o veľkosti 5. Premennú T_d zo vzorca 3.11 je možné nastaviť podobne ako pevný prah.

Neskôr som do detektora pridal metódu, ktorá počíta rozdiel 2 snímok pomocou korelácie týchto snímok. Metódu som nazval COR. Funkčnosť som implementoval podľa vzorca 3.6. Výstupom tejto funkcie je, podobne ako pri porovnaní histogramov, hodnota v rozmedzí 0 – 6000.

Ako posledné som do detektora implementoval porovnávanie s využitím kompenzácie pohybu, ktorá je počítaná z detekovaných hrán. Na výpočet kompenzácie som použil tri metódy z knižnice OpenCV. Prvou použitou metódou bola `cvCalcOpticalFlowLK`, ktorá hľadá hrany v snímke pomocou algoritmu Lucas–Kanade. Táto funkcia vypočíta pre každý pixel odhad jeho pohybu. Podobné vysledky vracia aj metóda `cvCalcOpticalFlowHS`. Tá na detekciu hrán využíva algoritmus Horn–Schunck. Metóda `cvCalcOpticalFlowBM` má odlišný výstup. Písmená BM vyjadrujú Block Matching, čo znamená, že výstom metódy je kompenzácia pohybu celých blokov. Na rozdiel od predchádzajúcich metód, táto počíta kompenzáciu pre bloky pixelov, takže implementácia do porovnávajúcich metód detektora je náročnejšia, ako v prípade `cvCalcOpticalFlowLK` a `cvCalcOpticalFlowHS`.

Do detektora som teda pridal metódu, ktorá sa stará o detekciu hrán a výpočet kompenzácie pohybu. Táto metóda sa volá pred porovnávaním 2 snímok, takže v čase porovnávania snímok sú vektory kompenzácie už známe. Pri porovnávaní sa zavolá zvolená metóda a v nej sa podľa argumentov pri spustení porovnáva s kompenzáciou, alebo bez nej. Všetky tri výpočty kompenzácie pohybu som implementoval do metód SAD, SSD, COR. Výpočty LK a HS som implementoval aj do vlastnej metódy BX.

4.4 Príprava programu na hromadné testovanie

Aby som mohol svoj detektor dostatočne otestovať, potreboval som, aby program reagoval na širokú škálu argumentov. Pri detekovaní strihov s pevne nastaveným prahom je potrebné testovať účinnosť detektora pri rôznych prahoch. Preto som zvolil ako argument pri spustení programu možnosť zadať počiatočný prah, konečný prah a krok. Pre účely testovania a vyhodnocovania výsledkov som vložil do detektora triedu `Sim`.

Trieda nastaví prah od počiatočného po koncový a volá triedu `Videocapture`. Po skončení detekovania načíta anotáciu k videu a vyhodnotí správnosť výstupu oproti anotácii. K testovaným videám sú k dispozícii anotácie vo forme XML súborov. Pre načítanie anotácie som využil knižnicu TinyXML [11].

Vústupy triedy `Sim` sú generované do zložky `plot`. Výstupmi sú štatistické údaje o detekcii v testovanom videu. Vygenerovaný súbor [názov súboru].dis obsahuje údaje o vypočítaných mierach odlišnosti medzi jednotlivými snímkami. Súbor [názov súboru].roc obsahuje údaje pre vykreslenie grafu, ktorým sa vyhodnoťuje úspešnosť detektora. Vytváranie názvov týchto súborov je popísané v manuáli k programu, ktorý je v prílohe B.

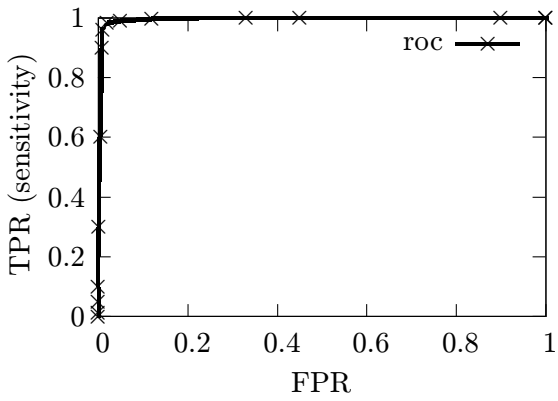
Na vyhodnotenie úspešnosti detektora sa využíva ROC krivka. Charakteristika ROC (Receiver operating characteristic) je vyjadrená krivkou závislosti TPR na FPR.

TPR (True positive rate, Sensitivity) vyjadruje citlivosť detektora. V našom prípade je to podiel správne detekovaných strihov na celkovom počte strihov. $TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$ kde TP (True Positives) predstavuje počet strihov, ktoré detektor správne detekoval, P (Positives) je celkový počet strihov vo videu a FN (False Negatives) je počet strihov, ktoré detektor označil ako nestrih.

FPR (False positive rate, 1 - specificity) vujadruje presnosť detektora. Pri detekovaní strihov označuje FPR podiel nesprávne detekovaných strihov v závislosti na celkovom počte prechodov, na ktorých nieje strih. $FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$ kde FP (False Positives) predstavuje počet prechodov, ktoré detektor označil ako strih, ale strih tam nebol a N (Negatives) predstavuje počet nestrihov. TN (True Negatives) predstavuje počet správne označených prechodov, kde sa nenachádza strih.

Na rozlíšenie úspešnosti sa počíta integrál krivky tohto grafu, označovaný tiež AUC (Area Under the Curve). Ideálny integrál je 1, čo znamená, že všetky strihy vo videu boli správne detekované a žiadny strih nebol vynechaný. Integrál menší ako 0.5 značí detektor, ktorý detekuje nesprávne viac ako polovicu prechodov.

Ak máme hodinové video s 25 snímkami za sekundu, tak toto video má približne 90 000 prechodov. Predstavme si, že je v ňom 333 strihov. Pri prahu, kedy detektor zachytí všetky strihy, detekuje prbližne 1000 prechodov nesprávne, a označí ich za strih (FN). Pri prahu, kedy detektor detekuje správne všetky nestrihy, nájde len približne 70 strihov, čo znamená, že 263 strihov označí ako nestrihy (FP). V bode, kde dosahuje najmenší počet nesprávne detekovaných prechodov môže byť napríklad 30 FP a 30 FN. Najmenšia chyba tohto detektora by sa mohla vypočítať ako $E_{cl} = \frac{FP+FN}{n} = \frac{60}{90000} = 0.000\overline{66}$. Pri dosadení do grafu ROC a následnom vypočítaní integrálu, by AUC vyšlo približne 0.999994. Na predstavivosť môže poslúžiť obrázok 4.1.

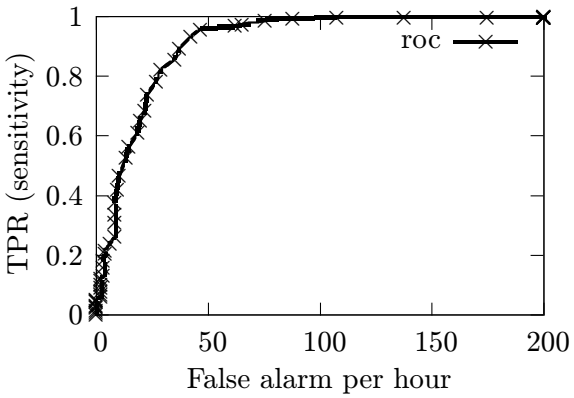


Obrázek 4.1: Graf ROC

Podľa grafu sa zdá, že krivka kopíruje ľavú a hornú stenu grafu a len na malom úseku v ľavom hornom rohu je graf zaoblený a odchyľuje sa od ideálu. Krivka sa od stien grafu oddelí už na začiatku ľavej a približne v polovici hornej steny grafu, toto však voľným okom nieje viditeľné. Aby som predišiel nepresnostiam pri meraní, a aby výsledky testovania boli na prvý pohľad viditeľné, upravil som ROC krivku. Na osi y som ponechal citlivosť TPR, ale na os x som vyniesol počet nesprávne detekovaných strihov na hodinu videa, teda 90 000 snímok. Viac ako 200 nesprávne detekovaných strihov za hodinu ma už nezaujíma. Preto som graf upravil tak, aby bolo na x – ovej osi najvyššie číslo 200. AUC vypočítam tak, že integrál tohto grafu vydelím najvyšším číslom na osi x, čo je 200. Novoupravený graf testovania tej istej metódy bude vyzerať podobne, ako na grafe 4.2.

Hodnotu na osi x v grafe vypočítam podľa vzorca $FA_{/h} = \frac{90000}{n} \cdot FP$, kde za n dosadím počet snímok vo videu.

Z grafu je na prvý pohľad možné zistiť, že sa nejedná o ideálny detektor a chybovosť je



Obrázek 4.2: Graf upravenej ROC

pomerne dosť vysoká. Podľa môjho názoru sú výsledky tohto grafu aj lepšie pochopiteľné. Je náročné predstaviť si význam FPR, ale keď vidím, že bolo detekovaných 50 nesprávnych strihov na hodinu videa, predstavuje sa to lepšie.

Kapitola 5

Testy

Na testovanie detektora som mal k dispozícii niekoľko anotovaných videí. Na stránke pána Laganièra [6] som našiel niekoľko vhodných videí aj s anotáciami. Z tejto stránky som použil dve videá, ktoré sú popísane na začiatku nasledujúcej sekcie tejto kapitoly. Pán Laganière však medzičasom tieto videá zo stránky odstránil.

Ostatné anotované videá, na ktorých som testoval svoj detektor som mal k dispozícii z dátového skladu pána Ing. Michala Hradiša. Bližší popis jednotlivých videí som zahrnul do nasledujúcej časti tejto kapitoly.

5.1 Popis testovaných dát

Najmenšie a najkratšie video, ktoré som použil pri testovaní má názov lisa.avi. Jedná sa o krátky úsek z animovaného seriálu. Video má rozlíšenie 144 × 192 pixelov a medzi 2631 snímkami obsahuje 7 strihov. Animácie v jednotlivých snímkach videa sú v niektorých pasážach dosť odlišné. Objekty sa tam pohybujú trhane, a pohyb objektov medzi dvomi sníkami je v niektorých záberoch dosť vysoký.

Ďalšie video, ktroré som prevzal od pána Laganièra má názov sexinthecity.avi. Jedná sa o ukážku zo seriálu. Video je v dobrej kvalite. Rozlíšenie videa je 272 × 336 pixelov. Video obsahuje 2631 snímok a 34 strihov. Toto video má zrejme najlepšie detekoavteľné strihy spomedzi testovaných videí.

Video 001.avi obsahuje dokument o veľrybách nahraný z televíznej stanice ČT1. Je natáčaný v podmorskom svete, takže všetky zábery majú nádych do modra. Z čoho vyplýva, že rozdelenie porovnávania na jednotlivé farby tu nebude mať veľký vplyv. Video sa skladá z 35147 snímok s rozlíšením 288 × 360 pixelov. Podľa anotácie obsahuje záznam 228 strihov.

Testovacie video z názvom 023.avi obsahuje záznam z hrania šachov na počítači. Vo väčšej časti videa je statický záber na šachovnicu a premiestňovanie figúriek. Vo videu sú len dva strihy a to strih zo šachovnice na komentátora a strih späť. Video je v rozlíšení 288 × 360 pixelov a obsahuje 2350 snímok.

Vo videu 025.avi je záznam z dokumentu vysielanom na ČT2. Záznam obsahuje množstvo záberov so približovaním a množstvo záberov zachytených v pohybe. Video sa skladá z 13125 snímok s rozlíšením 288 × 360 pixelov a obsahuje 93 strihov.

Na videu 029.avi je zachytené predvádzanie koňov na parkure. V zázname je niekoľko záberov na bežiaceho koňa s džokejom, v pozadí ktorého sa krajina pohybuje veľmi rýchlo. Rozlíšenie tohto videa je 288 × 360 pixelov a medzi 28184 snimkami je 82 strihov.

Najdlhšie testované video s názvom BG_12460.avi zachytáva starý dokument, zrejme z

čias začiatkov farebného filmu. Video je nekvalitné, zašumené a má slabý farebný kontrast. Skladá sa z 87027 snímok s rozlíšením 288 × 352 pixelov a obsahuje 333 strihov.

Posledné z testovaných videí má názov BG_3273.avi. Toto video je v dobrej kvalite, so silným farebným konrastom. Video obsahuje na začiatku množstvo zákerných efektov, ktoré určite množstvo detekujúcich metód zmätú. Myslím, že efekt, v ktorom sa približne tretina obrazu na pár sekúnd zafarbí do súvislej farby, budú detekovať ako strih všetky použité metódy. Video má rozlíšenie 288 × 352 pixelov, skladá sa z 36123 smímok a obsahuje 232 strihov.

Pre zhrnutie vyššie popísaných vlastností testovaných videí som vytvoril tabuľku 5.1

č.	**názov**	**výška (px)**	**šírka (px)**	**počet snímok**	**počet strihov**
1	lisa	144	192	650	7
2	sexinthecity	272	336	2631	34
3	001	288	360	35147	228
4	023	288	360	2350	2
5	025	288	360	13129	93
6	029	288	360	28184	82
7	BG_12460	288	352	87027	333
8	BG_3273	288	352	36123	232

Tabulka 5.1: Vlastnosti testovaných videí

Tým, že otestujem každú metódu na každom videu, získam prehľad o tom, ktorá metóda je lepšia pre ktoré video. Keďže chcem mať prehľad aj o celkových výsledkoch jednotlivých metód, rozhodol som sa vytvoriť pre každú metódu osobitný graf ROC. Do tohto grafu som vložil výsledky zo všetkých testovaných videí pre kontkrétnu metódu. Pri každom testovaní dostanem výsledky hodnôt TP, FP, TN a FN v závislosti na nastavenom prahu. Keď sčítam tieto hodnoty pre všetky videá, dostanem pre každú metódu zjednotený graf ROC. Celkové FPR bude teda

$$FPR = \frac{\sum_{i=1}^{n} TP_n}{\sum_{i=1}^{n} (TP_n + FN_n)}$$

a celkové FA

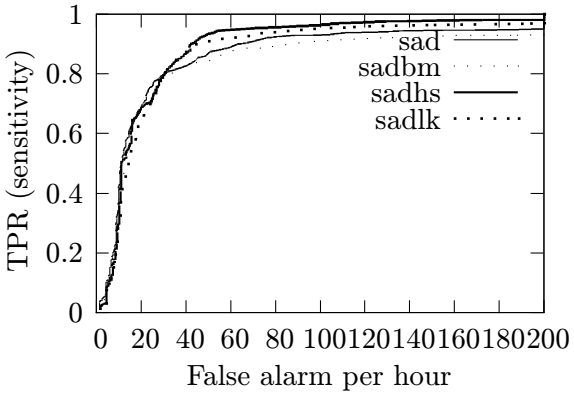
$$FA_{/h} = \frac{90000}{m} \cdot \sum_{i=1}^{n} FP_n,$$

kde n je počet testovaných videí a m je súčet počtu snímok všetkých testovaných videí.

5.2 Výsledky a porovnanie metód s nastavením pevného prahu

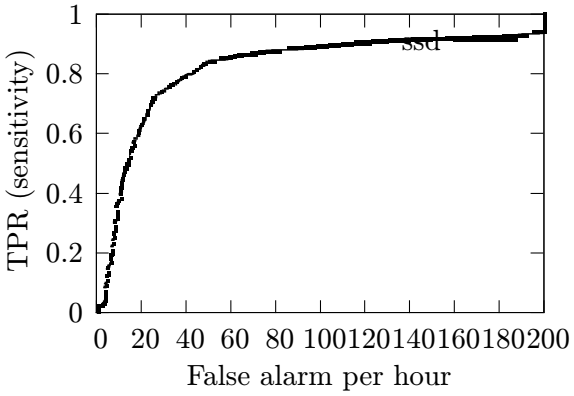
Ako prvú metódu som testoval SAD. Na testovanie tejto metódy som použil prah nastaveý v rozmedzí od 0 do 1500. Aby bol graf presný, zvolil som počet krokov 200, takže krok som zadal 7.5. Táto metóda si poradila s videami číslo 1, 2 a 4, kde AUC bolo 1. Vo videu 3 sa ukázalo, že porovnávanie po pixeloch je málo citlivé, ak je video málo kontrastné a prevažuje v ňom jedna farba vypočítané AUC tu bolo len 0.643. Ako ukázali výsledky detekcie vo videu 5, táto metóda má problém z rýchlymi pohybmi, výsledky však boli lepšie ako vo videu 3. V ostatných videách boli výsledky celkom slušné, AUC bolo väčšie ako 0.9. Celkové AUC tejto metódy som vypočítal na 0.8532. Lepšie výsledky dosiahla metóda s použitím kompenzácie pohybu. Použitie kompenzácie počítanej pre bloky pixelov

sa neosvečilo. Zrejme nahradzovanie celých blokov zmenšuje rozdiely medzi odlišnosťami strihov a nestrihov. Najlepšie výsledky dosiahla metóda s použím kompenzácie pohybu počítanej pomocou algoritmu HS. Prínos kompenzácie sa prejavil najmä na videách kde je veľa pohybu. Priebeh ROC pre všetky 4 varianty metódy SAD je na obrázku 5.1.



Obrázek 5.1: ROC metódy SAD

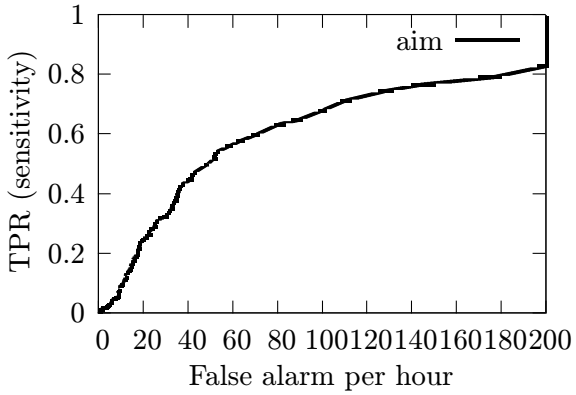
Metódu SSD som testoval v rozmedzí 0 – 250, s krokom 1.25. Vo videách 1, 2 a 4 bola metóda úspešná podobne ako SAD. Vo videu 5 boli výsledky o niečo lepšie. Očakával som, že vylepšené SAD bude mať aj lepšiu citlivosť na strihy, ale výsledky testovania ma sklamali. Vo všetkých ostatných videách boli vysledky slabšie ako s použitím SAD. Celkové výsledky metódy sú teda tiež slabšie. Vypočítané AUC je 0.816. Pretože táto metóda má podobné vlastnosti ako SAD, ale dosahovala slabšie výsledky, vynechal som testy na varianty s kompenzovaným pohybom. Priebeh ROC tejto metódy je zachytený na obrázku 5.2.



Obrázek 5.2: ROC metódy SSD

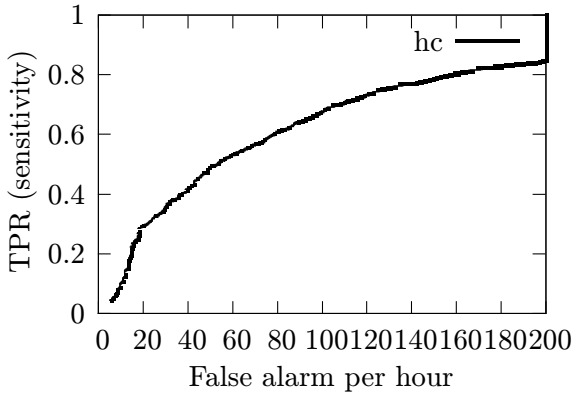
Pre testovanie metódy AIM som zvolil rozmedzie prahu na 0 – 350 s krokom 1.75. Vďaka tomu, že metóda je málo citlivá na pohyb, boli vo videu 5 lepšie vysledky ako v predchádzajúcich metódach. Vo videu 2 je niekoľko strihov, kde sú zmenené pozície objektov, ale celková intenzita snímok sa zmení len v malej miere. Toto sa podpísalo aj na slabších výsledkoch z tohto videa. Metóda mala slabé výsledky vo videách 4 a 7 kde je nízky farebný

kontrast a taktisto aj malé zmeny intenzity medzi snímkami. Výsledné AUC cez všetky videá je len 0.5882. Priebeh krivky ROC je na obrázku 5.3.



Obrázek 5.3: ROC metódy AIM

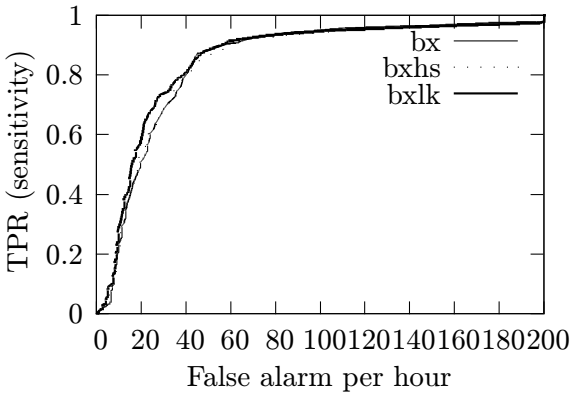
Metódu porovnania histogramov som testoval v rozmedzí 0 – 3500 s krokom 17.5. Testy ukázali, že histogram má dobré výsledky pri rýchlom pohybe vo videu 6. Rozdelenie histogramov na jednotlivé farby sa ukázalo ako neefektívne vo videách 3 a 7, kde je slabý farebný kontrast. Pri testovaní modrotónového videa č.3 boli výsledky dokonca slabšie ako výsledky metódy AIM. Celkovo má však táto metóda výsledky lepšie ako AIM, ale horšie ako SAD a SSD. Celkové AUC tejto metódy je 0.6014. Priebeh krivky ROC je patrný z obrázku 5.4.



Obrázek 5.4: ROC metódy HC

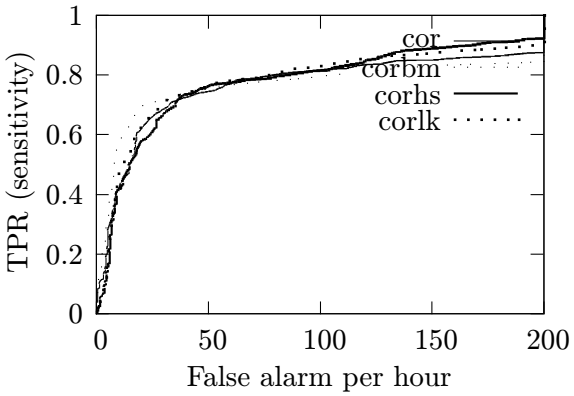
Na testovanie metódy BX som použil nastavenie prahu v rozmedzí od 0 do 2000 s krokom 10. Aj keď v prvom videu mali modifikácie s kompenzáciou lepšie výsledky ako varianta bez kompenzácie, ani tie nezachytili všetky strihy správne. Metóda mala vynikajúce výsledky vo videu 5, s pridaním kompenzácie pohybu boli dokonca zahytené správne všetky strihy a nestrihy. Tu sa ukázalo, že táto metóda je vhodná na dokumenty v dobrej kvalite, kde zábery obsahujú veľa približovaní objektov. S pridaním porovnania s kompenzáciou pohybu

sa v tejto metóde všetky výsledky okrem videa 3 vylepšili. Vypočítané AUC po testoch metódy BX je 0.8427, čo je v porovnaní s ostatnými metódami veľmi slušný výsledok. Podľa výsledkov testov je metóda BX a jej varianty na druhom mieste, hneď po metóde SAD. Priebeh ROC krivky tejto metódy a jej variant s kompenzovaným pohybom je znázornený na obrázku 5.5.



Obrázek 5.5: ROC metódy BX

Metódu krížovej korelácie (COR) som testoval v rozmedzí 0 – 3500 s krokom 17.5. S prvým videom si poradila jedine metóda CORHS, ktorá však mala celkový výsledok horší ako CORLK. Metóda si podobne, ako všetky ostatné metódy, najhoršie poradila s videom 3. Veľkú odlišnosť medzi jednotlivými variantami tejto metódy som nezaznamenal. Najlepšie AUC mala varianta CORLK, a to 0.7787. Graf ROC variant tejto metódy je možne vidieť na obrázku 5.6.



Obrázek 5.6: ROC metódy COR

V tabuľke 5.2 je podrobná sumarizácia výsledkov jednotlivých metód. Každý riadok tabuľky obsahuje názov testovanej metódy a výsledné AUC. AUC1 značí AUC vypočítané po testovaní videa číslo 1. AUC2, 3, 4, 5, 6, 7 a 8 podobne označujú AUC vypočítané pre konkrétne videá. AUC bez čísla je celkové AUC vypočítané cez všetky videá.

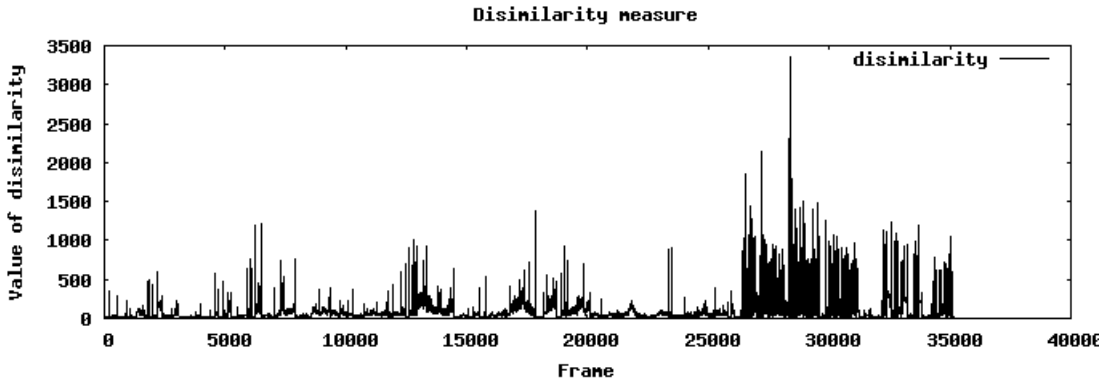
metóda	AUC	AUC1	AUC2	AUC3	AUC4	AUC5	AUC6	AUC7	AUC8
SAD	0.8532	1	1	0.643	1	0.83	0.903	0.93	0.918
SAD_{BM}	0.8311	1	1	0.629	1	0.92	0.86	0.915	0.88
SAD_{HS}	0.8816	1	1	0.668	1	0.95	0.92	0.93	0.924
SAD_{LK}	0.8665	1	1	0.666	1	0.92	0.94	0.92	0.92
SSD	0.816	1	1	0.58	1	0.837	0.79	0.923	0.858
AIM	0.5882	1	0.97	0.45	1	0.926	0.7	0.633	0.61
HC	0.6014	1	1	0.267	0.904	0.778	0.94	0.66	0.71
BX	0.8427	0.857	1	0.62	1	0.9989	0.87	0.93	0.918
BX_{HS}	0.8402	0.88	1	0.618	1	1	0.877	0.911	0.899
BX_{LK}	0.8503	0.88	1	0.606	1	1	0.927	0.91	0.913
COR	0.7614	0.88	1	0.56	1	0.986	0.94	0.91	0.757
COR_{BM}	0.7617	0.857	1	0.51	1	0.976	0.943	0.87	0.803
COR_{HS}	0.7721	1	1	0.56	1	0.996	0.976	0.84	0.78
COR_{LK}	0.7787	0.88	1	0.583	1	0.988	0.959	0.872	0.815

Tabulka 5.2: Výsledky testov jednotlivých metód pri testovaní s pevne nastaveným prahom

5.3 Výsledky a porovnanie metód s adaptívnym nastavením prahu

Vďaka tomu, že nastavenie testovacej premennej pri testoch s adaptívnym prahom nieje závislé na použitej metóde, mohol som zvoliť pre všetky testy jednotný rozsah. Všetky videá a všetky metódy som testoval s rozsahom hodnôt 0 – 100. Krok testu som zvolil 0.5.

Vďaka adaptívne nastavenému prahu si metóda SAD poradila veľmi dobre aj s videom 3. Toto video má slabý farebný kontrast. Vo videu sú preto vypočítané odlišnosti pomerne malé. Ako je patrné z obrázku 5.7, ku koncu videa sú odlišnosti vysoké.

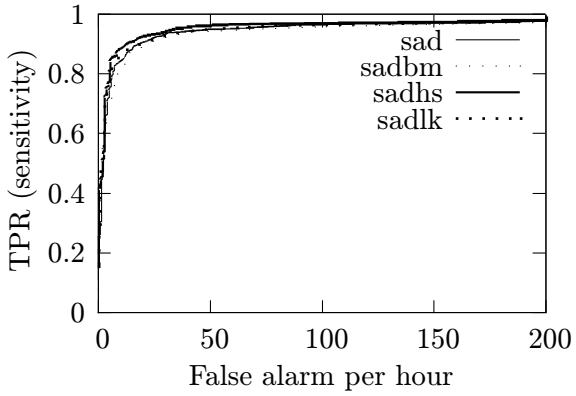


Obrázek 5.7: Graf miery odlišnosti medzi dvomi po sebe nasledujúcimi snímkami v závislosti na pozícii vo videu, vypočítaný z videa 3

Na konci videa sú nestrihové odlišnosti medzi snímkami dokonca vyššie, ako strihové odlišnosti v prvej polovici videa. V takomto videu sa naplno ukáže sila adaptívneho nastavenia prahu.

Metóda SAD mala vo všetkých testovaných videách AUC vyššiu ako 9, čo je veľmi slušný

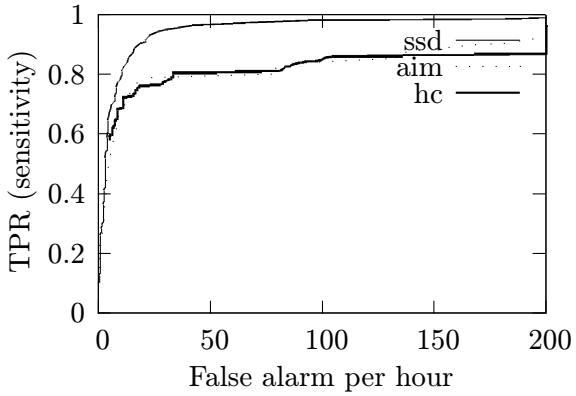
výsledok. Rozdiely medzi základnou metódou a metódou s použitím kompenzácie pohybu boli minimálne. Ako je patrné z grafu 5.8, najlepšie výsledky mala varianta s kompenzáciou pohybu pomocou HS. Základná metóda však na rozdiel od variant s kompenzovaným pohybom zachytila všetky strihy vo videu 5.



Obrázek 5.8: ROC metódy SAD

Výsledky metódy SSD sú podobne ako pri testovaní s pevným prahom slabšie, ako v prípade metódy SAD. Metóda mala lepšie výsledky ako SAD len pri testovaní videa 7. Aj tejto metóde sa s pomocou adaptívneho prahu podarilo správne detekovať všetky strihy vo videu 5. Celkové AUC je v tomto prípade 0.9461. Priebeh krivky ROC je znázornený na grafe 5.9.

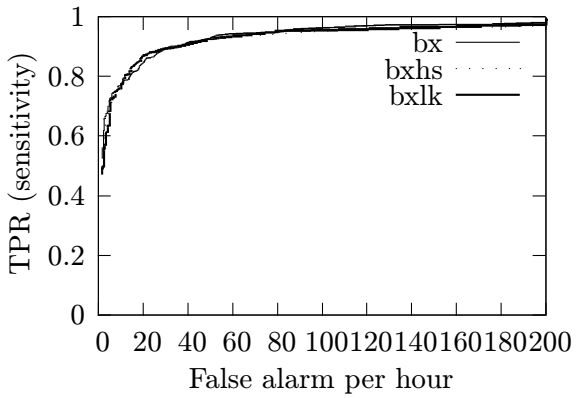
Metódy AIM a HC mali lepšie výsledky ako pri pevnom prahu. Ich celkové AUC sa pohybuje okolo 0.8, takže stále ostávajú najslabšími testovanými metódami. Obidve metódy si pohoršili oproti predchádzajúcim testom vo videu 7. Metóda HC mala problém s videom 1, kde adaptívny prah na rozdiel od pevného prahu neuspel. Pre porovnanie s metódou SSD som priebeh ich ROC kriviek znázornil na grafe 5.9.



Obrázek 5.9: ROC metódy AIM, HC a SSD

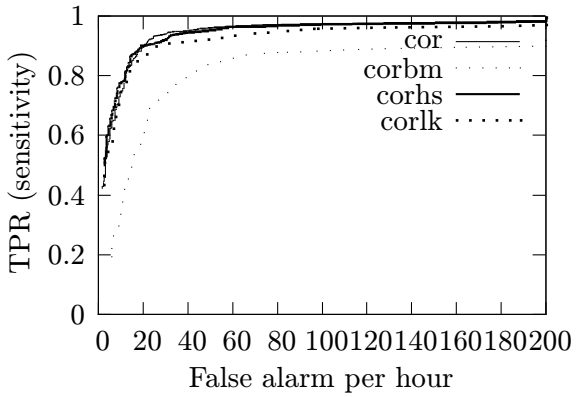
Celkové výsledky metódy BX sú lepšie ako v predchádzajúcom teste, ale zlepšenie nebolo až také veľké, ako u ostatných metód. Na rozdiel od pevného prahu, pri adaptívnom

prahu mala najlepšie výsledky základná metóda, bez použitia kompenzácie pohybu. Rozdiely medzi jednotlivými variantami metódy sú nepatrné. AUC základnej varianty metódy BX je 0.9252. Minimálne rozdiely medzi variantami sú patrné aj z grafu 5.10.



Obrázek 5.10: ROC metódy BX

Na rozdiel od výsledkov testov s pevným nastavením prahu mala s adaptívnym prahom najlepšie výsledky korelácie varianta bez kompenzácie pohybu. Varianta BM mala z testovania metódy COR najslabšie výsledky. Ak záber obsahuje rýchle zoomy, alebo zábery na pohyblivé objekty, počítanie kompenzácie pohybu po blokoch sa ukázalo byť nepresné. Táto varianta mala najhoršie výsledky vo videu 6, kde jed dosť častým záber na pohybujúceho sa džokeja. Veľmi slabé výsledky mala táto varianta aj pri testovaní videa 5, kde ja najčastejším záberom priblíženie a oddialenie nejakého statického objektu. ROC variant metódy COR je možné porovnať na grafe 5.11.



Obrázek 5.11: ROC metódy krížovej korelácie

V tabuľke 5.3 je podrobná sumarizácia výsledkov jednotlivých metód, ktoré som testoval s použitím adaptívneho nastavenia prahu.

metóda	**AUC**	**AUC1**	**AUC2**	**AUC3**	**AUC4**	**AUC5**	**AUC6**	**AUC7**	**AUC8**
SAD	0.9391	1	1	0.927	1	1	0.986	0.927	0.934
SAD_{BM}	0.9317	1	1	0.938	1	0.944	0.988	0.937	0.927
SAD_{HS}	0.9506	1	1	0.945	1	0.9992	0.987	0.941	0.932
SAD_{LK}	0.9425	1	1	0.935	1	0.9996	0.987	0.927	0.930
SSD	0.9461	1	1	0.911	1	1	0.986	0.954	0.909
AIM	0.8156	1	0.995	0.893	1	0.967	0.953	0.632	0.846
HC	0.8040	0.857	1	0.885	1	1	0.958	0.576	0.837
BX	0.9252	1	1	0.905	1	1	0.987	0.870	0.912
BX_{HS}	0.9166	0.88	1	0.907	1	1	0.987	0.874	0.872
BX_{LK}	0.9198	0.88	1	0.909	1	1	0.987	0.853	0.913
COR	0.9370	0.88	1	0.884	0.809	1	0.979	0.966	0.888
COR_{BM}	0.8063	0.857	1	0.888	1	0.277	0.073	0.988	0.899
COR_{HS}	0.9351	0.763	1	0.901	0.809	1	0.988	0.952	0.881
COR_{LK}	0.9154	0.738	1	0.899	809	1	0.98	0.900	0.882

Tabulka 5.3: Výsledky jednotlivých metód testovaných s adaptívne nastaveným prahom.

Kapitola 6

Záver

Cieľom tejto práce bol návrh a implementácia systému na detekciu strihov. Systém som implementoval po preštudovaní používaných techník na detekciu strihov. Mojim hlavným cieľom bolo vybrať metódu, ktorá by dokázala za akýchkoľvek podmienok správne detekovať strihy. Ďalším cieľom bolo priniesť vlastný algoritmus, ktorý by sa dokázal účinnosťou vyrovnať ostatným algoritmom.

Nepodarilo sa mi nájsť algoritmus, ktorý by bol na 100% úspešný vo všetkých videách. Zaujímavým zistením bolo, že rôzne „vylepšenia“ základných metód môžu vlastnosti základnej metódy podstatne zhoršiť. Použitie metód vylepšených o kompenzáciu pohybu objektov bolo určite prínosom. Jednoznačne najúčinnejšou bola metóda počítajúca sumu absolútnych rozdielov pixelov (SAD) v súčinnosti s kompenzáciou pohybu pomocou algoritmu Horn-Schunck. Jedinou slabinou metódy SAD je vysoká citlivosť na pohyb. Túto citlivosť dokázalo vo vysokej miere odstránit práve použite kompenzácie pohybu.

Metódy porovnávajúce celkové intenzity snímok boli v literatúre popisované ako vylpšenie tejto metódy. Možno vylepšujú problém s pohybom, ale ostatné nedostatky prevyšujú toto vylepšene.

Použitie adaptívneho prahu je ďalším vylepšením detektora, ktoré sa ukázalo ako mimoriadne prínosné. Vďaka tejto možnosti sa zlepšili výsledky všetkým testovaným metódam.

Algoritmus, ktorý som navrhol ako vylepšenie algoritmu SAD, priniesol lepšie výsledky len u jedného testovaného videa. Ostatné výsledky však boli horšie. Takže vylepšiť tento algoritmus sa nepodarilo ani mne.

Detektor strihov je len jednou časťou systému, umožňujúceho vytváranie videoarchívov. Vhodným pokračovaním práce by mohol byť vývoj programu, ktorý by dokázal video rozdeliť na zábery a tieto zábery osobitne ukladať do videoarchívu. Myslím, že s pridaním grafického užívateľského rozhania by program našiel využitie u viacerých amatérskych kameramanov. Z niekoľkými strihacími programami som už pracoval, ale ešte som sa nestretol s programom, ktorý by toto dokázal. Niektoré dokážu rozdeliť záznam z kamery, podľa časových značiek, ale nedokážu automaticky uložiť jednotlivé zábery do osobitných súborov na disku. Takýto program by určite ušetril amatérskemu kameramanovi mnoho času.

Ďalšou možnosťou vylepšenia práce je pridanie detekcie postupných prechodov medzi zábermi. V poslednej dobe sa často v amatérskych videách objavuje množstvo efektov, takže táto úprava by tiež našla využitie.

Literatura

[1] Boreczky, J. S. a Rowe, L. A. J. Electronic Imaging 05(02), 122-128, Edward R. Dougherty; Ed. In *Proceedings of the second ACM international conference on Multimedia* [online]. Berkeley: [b.n.], 1996 [cit. 18. dubna 2010]. Dostupné na: http://spie.org/x648.html?product_id=234794.

[2] Bradski, G. a Kaehler, A. *Learning OpenCV*. 1. vyd. Sebastopol: O'Reilly Media, Inc., 2008. ISBN 978-0-596-51613-0.

[3] Chávez, G. C., Cord, M., Philipp Foliguet, S. et al. Robust Scene Cut Detection by Supervised Learning. In *EUSIPCO* [online]. Firenze, Italy: [b.n.], 2006 [cit. 10. dubna 2010]. Dostupné na: http://publi-etis.ensea.fr/2006/CCPPD06/.

[4] Hampapur, A., Jain, R. a Weymouth, T. Digital Video Segmentation. In *Proceedings of the second ACM international conference on Multimedia* [online]. Ann Arbor: [b.n.], 1994 [cit. 18. dubna 2010]. Dostupné na: http://portal.acm.org/citation.cfm?id=192593.192699.

[5] Jun, S. a Park, S. *An Automatic Cut Detection Algorithm using Median Filter and Neural network*. [b.m.]: Han Yang University.

[6] Laganiere, R. *Computer Vision and Image Processing* [online]. [cit. 10. dubna 2010]. Dostupné na: http://www.site.uottawa.ca/~laganier/.

[7] Nagasaka, A. a Tanaka, Y. Automatic Video Indexing and Full-Video Search for Object Appearances. In Knuth, E. a Wegner, L. M. (ed.). *Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems II*. Amsterdam: North-Holland Publishing Co., 1992. S. 113–127. ISBN 0-444-89609-0.

[8] *OpenCV 2.1 C Reference* [online]. [cit. 10. dubna 2010]. Dostupné na: http://opencv.willowgarage.com/documentation/c/index.html.

[9] Porter, S. V., Mirmehdi, M. a Thomas, B. T. Video Cut Detection using Frequency Domain Correlation. In *15th International Conference on Pattern Recognition (ICPR'00) - Volume 3* [online]. Bristol: [b.n.], 2000 [cit. 10. dubna 2010]. Dostupné na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.4593&rep=rep1&type=pdf.

[10] Shen, B. *HDH Based Compressed Video Cut Detection* [online]. Palo Alto: Visual Computing Department, HP Laboratories [cit. 10. dubna 2010]. Dostupné na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.8974&rep=rep1&type=pdf.

[11] *Simple XML* [online]. [cit. 10. dubna 2010]. Dostupné na: http://www.grinninglizard.com/tinyxml/.

[12] Tahaghoghi, S. M. M., Williams, H. E., Thom, J. A. et al. Video Cut Detection using Frame Windows. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38* [online]. Hamilton: [b.n.], 2005 [cit. 10. dubna 2010]. Dostupné na: http://portal.acm.org/citation.cfm?id=1082161.1082183.

[13] Wikimedia Foundation, Inc. *Wikipedia* [online]. [cit. 10. dubna 2010]. Dostupné na: http://www.wikipedia.org.

[14] Yusoff, Y., Christmas, W. a Kittler, J. *Video Shot Cut Detection Using Adaptive Tresholding* [online]. Guildford: Centre for Vision, Speech and Signal Processing, University of Surrey [cit. 10. dubna 2010]. Dostupné na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.7173&rep=rep1&type=pdf.

[15] Yusoff, Y., Christmas, W. a Kittler, J. A Study on Automatic Shot Change Detection. In *Multimedia Applications, Services and Techniques - ECMAST'98* [online]. 1998 [cit. 10. dubna 2010]. Dostupné na: http://www.springerlink.com/content/9483132370306150/.

Dodatek A

Obsah CD

- `anotation` – anotácie k priloženým testovacím videám
- `doc/pdf` – tento dokument vo formáte pdf
- `doc` – zdrojový kód tohto dokumentu v jazyku LaTeX
- `doc/fig` – zdrojové kódy k vykresľovaným grafom a obrázky tohto dokumentu
- `myout` – prázdna zložka, ktorá slúži na ukladanie dočasných súborov počas behu programu.
- `plot` – zložka obsahuje skript pre program gnuplot, pomocou ktorého sú vykresľované grafy z výstupných udajov. Do zložky sú ukladané výstupy programu.
- `src` – zdrojový kód projektu (NetBeans IDE 6.8)
- `video` – dva testovacie videá na ukážku

Dodatek B

Manuál k programu

Aby mohol detektor testovať komnbináciu rôznych metód, je potrebné pri spustení zadať niekoľko parametrov. Prvým parametrom je možné určiť, či detektor pracuje s pevným prahom, alebo s adaptívnym prahom. Parametrom `-stat` povieme detektoru, že chceme vypočítať štatistiky s nastavením pevného prahu. Ak namiesto neho použimeme parameter `-adt` program bude počítať štatistiky s použitím adaptívneho prahu. Program je tiež možné spustiť s parametrom `--help`, pomocou ktorého vypíšeme nápovedu k programu.

Aby bolo možné vizuálne overiť na videu správnosť detekcie, pridal som do detektora aj možnosť zobraziť video počas detekovania. Ak k jednému s týchto parametrov pripojíme `show` napríklad `-statshow`, tak budú počas výpočtu rozdielov zobrazené aktuálne počítané snímky. Túto funkčnosť som pridal hlavne kvôli počítaniu histogramov a hľadaniu hrán v snímkach. Pri zadaní funkcie výpočtu histogramov sa mi zobrazia pri snímkach aj ich histogramy a tak si možem vizuálne overiť, nakoľko sú histogramy odlišné. Ak majú byť počítané vektory pohybu, vďaka tejto funkčnosti sa mi zobrazí predpokladaná nasledujúca snímka a túto si môžem porovnať s reálnou nasledujúcou snímkou.

Ďalšia funkčnosť, ktorú som pridal na vizuálne overenie, sa spustí pridaním `showcuts`. Ako už názov napovedá, detektor zobrazí video a zastaví sa medzi snímkami, kde bol detekovaný strih. Pri zadaní tohto parametra detektor štandardne počíta rozdiely a zisťuje, kde sa nachádzajú strihy. Po skončení detekcie sa spustí video a ukážu sa 2 snímky. Pri každom detekovanom strihu sa video zastaví a zobrazia sa snímka pred strihom a snímka po strihu. Toto zastavené video je možné ďalej prehrávať stlačením ľubovoľnej klávesy. Ak je video dlhé a nechceme ho už ďalej prehrávať, stačí stlačiť klávesu ESC a program sa skončí. Zastavenie prehrávania videa je možné dosiahnuť stlačením klávesy p.

Ako prvý parameter musí byť teda požitá jedna z nasledujúcich možností: `-stat`, `-adt`, `-statshow`, `-adtshow`, `-statshowcuts`, alebo `-adtshowcuts`.

Druhý parameter určuje metódu, ktorá sa má použiť pri počítaní rozdielov medzi snímkami. Parameter sa zadáva bez úvodnej pomlčky. Názov parametru presne odpovedá názvu funkcie, ktorá sa má použiť, jediný rozdiel je vo veľkosti písmen. Do parametru sa funkcia zadáva s malými písmenami. Napríklad pre použiie AIM je to `aim`. Ak chcem použiť vektory, za názov funkcie pridám názov metódy, ktorou sa majú vektory hľadať. Výpis všetkých použiteľných parametrov je teda nasledovný: `aim`, `bx`, `cor`, `hc`, `sad` a `ssd`. A s použitím vektorov ešte pribudne `bxlk`, `bxhs`, `corlk`, `corhs`, `corbm`, `sadlk`, `sadhs`, `sadbm`, `ssdlk`, `ssdhs` a `ssdbm`.

Ako tretí parameter je potrebné zadať názov súboru obsahujúceho video. Toto video sa musí nachádzať v zložke video. Zložka, v ktorej sa nachádza skompilovaný program musí obsahovať 4 podzložky. Podzložka `anotation` obsahuje anotácie k jednotlivým videám.

Názov anotácie sa musí zhodovať s celým názvom videa + obsahuje príponu .xml. Takže ak mám video lisa.avi, tak musím mať anotáciu s názvom lisa.avi.xml. Do zložky `myout` sa počas behu programu zapisujú súbory, potrebné pre ďalšie spracovanie programu. V zložke `plot` je možné po skončení činnosti detektora nájsť štastiky o presnosti generovania doplnené grafom. Názov súboru so štatistikami a obrázku s grafom sa zostaví z názvu súboru obsahujúceho video, zo spracovávajúcej funkcie a z prvého zadaného parametru. Na vygenerovanom obrázku je možné nájsť graf s vypočítanou mierou odlišnosti pozdĺž celého videa, ďalej graf počtu chybných detekcií v závislosti na veľkosti prahu a nakoniec graf vlastnej ROC krivky. Poslednou potrebnou zložkou je `video`, kam je potrebné uložiť videá, ktoré sa majú spracovávať.

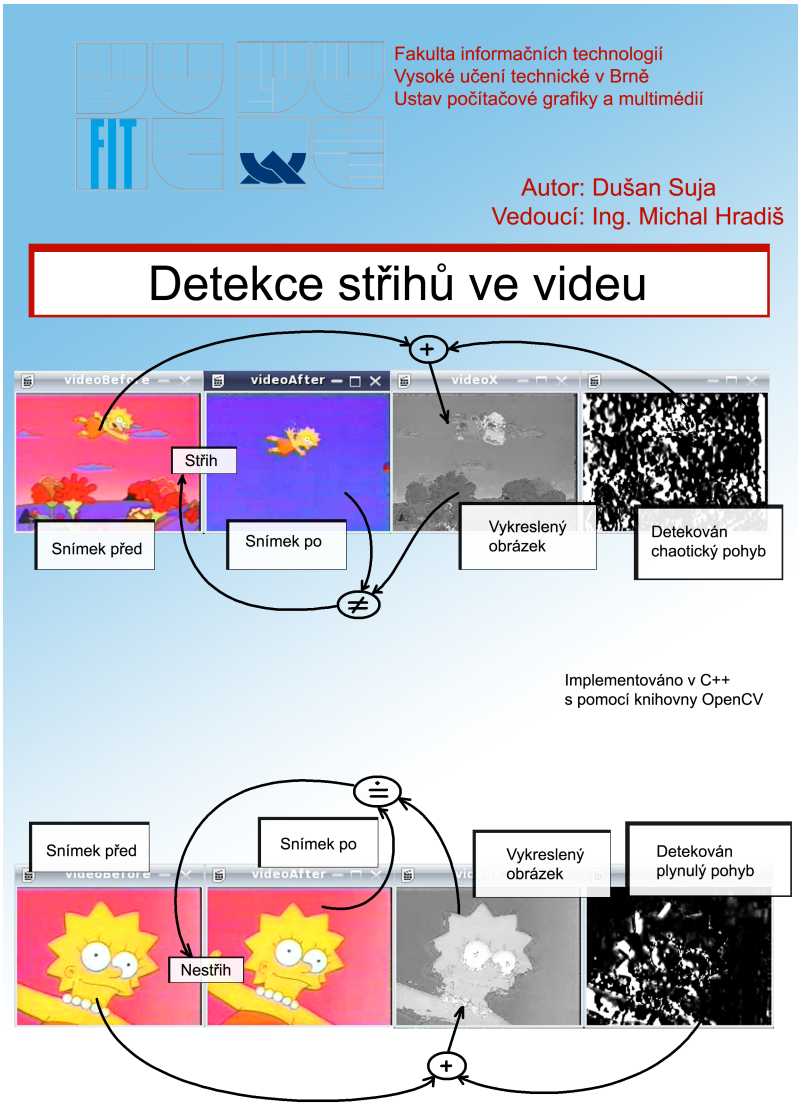
Štvrtý parameter sa používa dvomi spôsobmi. Ak program spustíme so štyrmi parametrami, do tohto parametru je možné zadať hodnotu prahu, prípadne hodnotu T_d pri počítaní adaptívneho prahu. Ak program spustíme so šiestimi alebo siedmimi parametrami, štvrtý slúži na zadanie počiatočného prahu, piaty na zadanie konečného prahu a šiesty na zadanie kroku. Môže sa stať, že anotácia k videu je posunutá o niekoľko snímok a preto by mohol byť detektor vyhodnotený ako nepresný, aj keď detekoval správne. Preto som pridal siedmy parameter, ktorým je možné túto chybu opraviť. Zadáva sa do neho o koľko snímok sa má program pri testovaní posunúť.

Príklad spustenia programu s priloženými testovacími videami:

- `./vcap -adt sadhs lisa.avi 5 30 1`
- `./vcap -adtshowcuts sadhs lisa.avi 20`
- `./vcap -stat aim lisa.avi 40 50 1`
- `./vcap -statshow sadhs lisa.avi 50`
- `./vcap -stat hc sexinthecity.avi 150 200 5 1`

Dodatek C

Plagát



Obrázek C.1: Náhľad plagátu, ktorý prezentuje túto prácu